Lattice Transcript Sponge

ccertain *

October 21, 2026

1 Sketch

A Fiat-Shamir transcript hash does two jobs: it absorbs every message of the proof, and it produces challenges that the prover cannot steer. Inside a circuit that verifies a lattice argument the two jobs have very different costs. Absorbing is the bulk of the work, proportional to the proof size, and an algebraic hash such as Poseidon [2] spends about 140 non-binary constraints per absorbed element. Challenges are few, about one per sumcheck round.

The sponge of this note splits the jobs. Absorption is an Ajtai hash [1] on bits: every element is decomposed into its $e = \lceil \log_2 q \rceil$ bits and folded into a state of κ field elements through a public matrix, which costs e binary constraints per element and is linear in the bits. Challenges are squeezed by running the state through a Poseidon permutation, so that no challenge is a linear function of anything the prover sent. Long challenge vectors are expanded from a Poseidon output by the lattice hash again. Collision resistance of the absorption reduces to SIS at ℓ_∞ norm 1, and the Fiat-Shamir argument reduces to the random oracle model for Poseidon plus that collision resistance.

The construction is not hiding and is not a commitment: the state is a linear function of the absorbed bits. Section 6 keeps an earlier note on the non-compressing case.

2 Construction

2.1 Parameters

- q prime, $e = \lceil \log_2 q \rceil$. Elements are decomposed through their canonical representatives in $[0, q)$, little-endian: $\mathrm{bits} : \mathbb{Z}_q \to \{0,1\}^e$, extended to vectors by concatenation. An implementation that stores elements in another form (Montgomery) converts first.
- κ, the state width in elements. κe has to exceed 256 bits and κ has to make the SIS instance below hard; $\kappa = 8$ is the working value for Mersenne61 (488 bits) pending a lattice estimate.
- R, the rate: elements absorbed per block. $R = 64$ is the working value; larger R lowers the share of the state re-decomposition, κ/R, and enlarges the matrix.
- $\mathbf{A} \in \mathbb{Z}_q^{\kappa \times e(\kappa+R)}$, uniformly random, expanded from a public seed and a label.

*ccertain@tutanota.com

- $P : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$, a Poseidon permutation of width $t \geq \kappa + 2$ over $\mathbb{Z}_q$ whose rate exceeds κ. For Mersenne61 the instance is $t = 24$ with 8 capacity lanes (488 bits) and the S-box x^{17}, the smallest exponent coprime to $q - 1$ since $3, 5, 7, 11$ and 13 divide it; $R_F = 8$, $R_P = 22$ from the hadeshash script at $M = 256$. The Milli test field uses the existing width 48 instance.
- $\mathbf{A}_{exp} \in \mathbb{Z}_q^{\kappa \times e(t+1)}$, a second public matrix for the expansion of seeds.

2.2 State

The sponge holds a lattice state $\mathbf{s} \in \mathbb{Z}_q^\kappa$, initially $\mathbf{0}$, a buffer of at most R pending elements, and a squeeze counter.

2.3 Absorb

Elements are appended to the buffer. When R elements are pending, or when a squeeze is requested, the block is compressed into the state:

$$\mathbf{s} \leftarrow \mathbf{A} \cdot \big(\mathrm{bits}(\mathbf{s}) \,\|\, \mathrm{bits}(m_1, \ldots, m_R)\big)$$

At a squeeze the buffer is padded unconditionally, pad10*: the element 1 is appended, opening a new block if the buffer was full, then zeros to the block boundary, and the block is compressed. A padded block therefore always ends in $1, 0, \ldots, 0$ after a sequence that is the whole buffer, and no two different buffers pad to the same block. Domain separation is by absorbing a label as elements before the messages of a phase, the `commit(label)` pattern of the transcript oracle.

2.4 Squeeze

A squeeze first compresses the padded pending block, then runs the state through Poseidon with the counter as a tag:

$$\mathbf{y} = P\big(\mathbf{s} \,\|\, ctr \,\|\, 0^{t-\kappa-1}\big) \in \mathbb{Z}_q^t, \qquad ctr \leftarrow ctr + 1$$

and absorbs $y_1, \ldots, y_t$ as a padded block, $\mathbf{s} \leftarrow \mathbf{A} \cdot (\mathrm{bits}(\mathbf{s}) \,\|\, \mathrm{bits}(\mathrm{pad}(\mathbf{y})))$. The challenge is read from the output lanes of $\mathbf{y}$, 8 elements for the Mersenne61 instance and 16 for Milli; a request for more is served by further squeezes. Re-absorbing $\mathbf{y}$ binds the squeeze into every later state, as the permutation does in a conventional sponge. An element of the cubic extension $\mathbb{F}_{q^3}$ is three squeezed elements.

2.5 Expand

Challenge vectors longer than a few elements, such as the $(\kappa' + 3)k$ coefficients of a random combination or the rk bits of an opening challenge matrix, are not squeezed from Poseidon. One squeeze produces a seed $\mathbf{y} \in \mathbb{Z}_q^t$, and the vector is the lattice hash of the whole seed in counter mode:

$$\mathbf{x}_i = \mathbf{A}_{exp} \cdot \big(\mathrm{bits}(y_1, \ldots, y_t) \,\|\, \mathrm{bits}(i)\big), \qquad i = 0, 1, \ldots$$

read κ elements per block. Every block takes the full te bits of the seed, which leaves $(t-\kappa)e$ bits of margin over the κe output bits (488 at the working parameters) and makes the blocks independent given the seed; a chain $\mathbf{x}_{i+1} = \mathbf{A}_{exp} \cdot \text{bits}(\mathbf{x}_i, i)$ would have only e bits of margin per block. Binary challenges are the low $e-1$ bits of each element, as in the SLIP argument, biased by at most $2^{e-1}/q$ per bit. The expansion does not touch the sponge state.

3 Security

3.1 Collision Resistance of the Absorption

Lemma 1. *Two different sequences of absorbed elements that reach the same lattice state yield a vector* $\boldsymbol{u} \in \{-1, 0, 1\}^{e(\kappa+R)}$, $\boldsymbol{u} \neq \boldsymbol{0}$, *with* $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{0}$.

Proof. Walk both sequences back from the common state. At the last block the inputs $\text{bits}(\mathbf{s})\|\text{bits}(\mathbf{m})$ and $\text{bits}(\mathbf{s}')\|\text{bits}(\mathbf{m}')$ either differ, which gives $\mathbf{u}$ as their difference, or agree, in which case the previous states and the last blocks agree, and the sequences differ earlier; induction on the number of blocks. The padding keeps a padded partial block distinct from every full block. □

So a collision is a solution of $\text{SIS}_{\kappa,e(\kappa+R),q}$ with ℓ_∞ norm 1 and ℓ_2 norm at most $\sqrt{e(\kappa+R)}$, about $2^{6.1}$ at the working parameters. κ is to be set with a lattice estimator for that instance. Second preimages are the same problem; preimages of a state are inhomogeneous SIS.

3.2 Fiat-Shamir

Model the squeeze function $\mathbf{s} \mapsto P(\mathbf{s} \,\|\, ctr \,\|\, 0)$ as a random oracle, which is the assumption a Poseidon transcript already makes (P itself is a permutation). Every challenge of the protocol is a function of a Poseidon output on the lattice state, so the transcript hash has the form $H = P \circ h$ with h the absorption. A prover that makes Q oracle queries and produces an accepting transcript with probability above the sum over the squeezes of the bad-challenge probabilities times Q (the usual Fiat-Shamir bound) has, for some squeeze, two transcripts with the same lattice state, since every query to P on a fresh state returns a fresh uniform challenge. Two transcripts with the same state are a collision in h. Therefore the Fiat-Shamir error is the random oracle bound plus Adv^{SIS} for $\mathbf{A}$. The linearity of h in the absorbed bits is harmless: it is never exposed to the prover as a challenge, only compressed into an oracle input.

This is why the state must not be read as a challenge directly. A challenge that is affine in the last absorbed bits can be steered by solving a short-vector problem whose difficulty depends on the shape of the bad set: for a bad set of one value it is inhomogeneous SIS, for a hyperplane it is a high-density knapsack in a few thousand variables, which is not a problem one wants to rely on.

3.3 Expansion

The seed of an expansion is a uniform oracle output outside the prover's control, so the probability that an expanded challenge vector is bad is a probability over seeds, $\Pr_{\mathbf{y}}[\text{Expand}(\mathbf{y}) \in \text{Bad}]$, and the prover can only re-roll seeds by changing the transcript, which the oracle bound already

counts. The expansion therefore needs no cryptographic property, only a balanced output: the fraction of seeds mapping into a bad set has to be close to the measure of the bad set. For the lattice expansion this is a heuristic, the same kind as modelling $\mathbf{A}_{exp}\mathbf{x}$ for binary $\mathbf{x}$ as uniform when the input has $te \gg \kappa e$ bits of entropy, which is why the expansion takes the whole seed in every block. A protocol whose bad sets are structured with respect to the bit extraction, e.g. aligned with the low bits of elements, should state the bias.

4 Cost

Inside a circuit, with the SLIP cost model where a binary constraint row costs 3 witness cells and a non-binary one $3e$:

	constraints	cells
absorb one element (lattice)	e binary	$4e \approx 244$
state re-decomposition per block	κe binary	$4\kappa e$, about 12% of a block at $R = 6$
absorb one element (Poseidon, for comparison)	≈ 140 non-binary	≈ 36000
one squeeze (Poseidon permutation)	≈ 2200 non-binary	$\approx 5.4 \cdot 10^5$
expand one element	e binary	$4e$

The products by $\mathbf{A}$ and $\mathbf{A}_{exp}$ and the recompositions are linear combinations with public coefficients, which cost nothing in an R1CS. Natively the absorption is column sums of $\mathbf{A}$ selected by bits, no multiplications; a transcript of N elements costs $Ne\kappa$ additions.

For a proof of N elements with S squeezes and X expanded elements the circuit cost is about $4eN + 5.4 \cdot 10^5 S + 4eX$ cells. A FLIP layer (`../flip/spec.pdf`) has $N \approx 6.5 \cdot 10^4$, $S \approx 18$ and $X \approx 1.6 \cdot 10^4$, about $3.2 \cdot 10^7$ cells against $2.3 \cdot 10^9$ with Poseidon absorbing everything.

5 Open Items

1. κ from a lattice estimator for $\mathrm{SIS}_{\kappa, e(\kappa+R), q}$ at ℓ_∞ norm 1.

2. The x^{17} S-box of the Mersenne61 Poseidon costs five multiplications per lane against three for x^5; a different permutation for this field is worth considering.

3. The implementation, `LatticeSponge`, follows the construction above behind the `Sponge` trait, with $\kappa = 8$, $R = 64$ and the absorbed length as the Poseidon tag. The transcript oracle builds a fresh sponge per squeeze from the previous Poseidon output and the messages since, which is the re-absorption of Section 2.4. `LatticeHashSponge` is the earlier placeholder, kept for fields without a Poseidon instance: it multiplies raw field elements without decomposition and is not collision resistant.

4. The balance of the expansion, Section 3.3, as a stated heuristic.

6 Note: Injectivity Without Compression

The following is the earlier content of this document. It concerns a square or nearly square $\mathbf{A}$ and does not apply to the compressing hash above, where collisions exist and collision resistance is computational.

Lattice problems admit statistical security in binding or hiding as described in [3]. The security of binding vs hiding is a tradeoff where computational security for both yields statistical security for neither. In the binding-only SIS case statistical security is based on second preimage non-existence for all short elements on the lattice: $\forall \mathbf{x} \not\exists \mathbf{x}' : \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}'$. Write the set of short elements $S_\beta = \{x \mid \|x\|_\infty \leq \beta\}$, $|S_\beta| = (2\beta+1)^n$. [3] argues that the family $h_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, $S_\beta^n \to \mathbb{Z}_q^n$, is a universal hash. One can instead bound the probability of selecting a non-injective $h_{\mathbf{A}}$:

$$\Pr_{\mathbf{A}}\Big[\exists \mathbf{x} \in S_{2\beta}^n \setminus \{\mathbf{0}\} : \mathbf{A}\mathbf{x} = \mathbf{0}\Big] \leq |S_{2\beta}|^n q^{-n}$$

For $\log_2 q \geq 16$, $\beta = 1$, $n \geq 10$ this is below 2^{-160}: a random square $\mathbf{A}$ is injective on short vectors with overwhelming probability, so no second preimages exist. There is no compression, so this is not a hash; it is a statistically binding, non-hiding encoding of short vectors.

References

[1] Miklós Ajtai, Generating Hard Instances of Lattice Problems, STOC 1996.

[2] Lorenzo Grassi and Dmitry Khovratovich and Christian Rechberger and Arnab Roy and Markus Schofnegger, Poseidon: A New Hash Function for Zero-Knowledge Proof Systems, USENIX Security 2021, Cryptology ePrint Archive, Paper 2019/458, `https://eprint.iacr.org/2019/458`

[3] Carsten Baum and Ivan Damgård and Vadim Lyubashevsky and Sabine Oechsner and Chris Peikert, More Efficient Commitments from Structured Lattice Assumptions, Cryptology ePrint Archive, Paper 2016/997, 2016, `https://eprint.iacr.org/2016/997`